May 2021

Reconciling Assets in a Rights Management Database

Thomas Bugnon

George Huang

Sha Chang

Xiangyu Li

Arthur Gaudriot

See next page for additional authors

Inventor(s)
Thomas Bugnon, George Huang, Sha Chang, Xiangyu Li, Arthur Gaudriot, and Keith Wedelich

Reconciling Assets in a Rights Management Database

ABSTRACT

A rights management database maps content to content owners. Such a database can have duplicates of essentially identical assets, a problem referred to as undermerging. Conversely, a single asset can erroneously include multiple distinct assets, a problem referred to as overmerging. This disclosure describes techniques to automatically resolve overmerged assets and undermerged assets in a rights management database. Per the techniques, a consistency signal for an asset is computed. A logic module uses the consistency signal and a strict-match signal to merge assets that correspond to the same content. Another logic module detects duplicate reference material and builds a graph of thus-far undermerged assets that can be merged. The techniques detect and resolve undermerged and overmerged assets at scale, correctly handling slightly transformed duplicates, e.g., remasters, trimmed versions, sped-up versions, etc.

KEYWORDS

- User-generated content (UGC)
- User-contributed content (UCC)
- Content matching
- Digital rights management (DRM)
- Copyright violation
- Infringing content
- Undermerging
- Overmerging
- Content metadata

BACKGROUND

Rights management databases that maintain a record of pre-existing rights (e.g., copyright) to creative works are utilized by various service providers, e.g., audio/video hosting/sharing services, games, and other apps/services that provide user-generated content

(UGC). Typically, such services enable users to upload any UGC which may at the time of upload or at a later point be checked against a rights management database to determine whether any segment of the UGC is potentially violative of pre-existing rights.

A rights management database maps content (also known as assets, that may include music or other audio, video, multimedia, etc.) to content owners. A rights management database may include duplicates of essentially identical assets, a problem referred to as undermerging. Conversely, a single asset can erroneously be a concatenation of multiple distinct assets, a problem referred to overmerging. Both undermerging and overmerging are data reconciliation problems; the presence of overmerged or undermerged assets degrades the quality of data in a rights management database.

Reduction of overmerges and undermerges to acceptable levels is important to ensure high quality of a rights management database. Asset-matching technologies thus far used in asset-management do not offer fine-tuning, leading to slightly transformed duplicates such as audio remasters, trimmed versions, etc., often being classified as separate assets.

DESCRIPTION

This disclosure describes techniques, based on an underlying, high precision-recall, asset-matching module, to automatically resolve overmerged assets and undermerged assets. The asset-matching module determines match between any given pair of assets based on the similarity of their audio or video signal. Per the techniques, the consistency of an asset is computed. An asset is considered to be consistent if the reference material associated with it corresponds to the same piece of intellectual property. This enables the determination of the state, e.g., good or bad, of the asset. The consistency check is performed based on strict-match annotation provided by the asset-matching module. While non-strict matching matches across a

variety of transforms typically found in user-generated content, e.g., background talking or disturbances, remixes, distortions, and other transformations, strict-match annotations are provided only when the matched content is near-identical, e.g., remasters, or speed differences. Asset consistency is described in greater detail below.

A logic module, referred to as the reference-flow module, receives incoming, e.g., newly uploaded, reference material. The reference-flow module operates through the lifecycle of all references, including looking up new ones in the match module and triggering merge attempts. The reference-flow module uses the consistency signal and the strict-match signal to merge assets that correspond to the same piece of intellectual property. Asset merging is described in greater detail below.

Another module, referred to as the rights-management module, detects duplicate reference material based on strict asset-matches and uses that information to build a graph of undermerged assets, which can then be merged. Metadata can include fields such as title, artist, etc., and can be provided by the asset uploader, the content owner, etc. and can include inaccuracies. Upon merging, the metadata of an asset, if inconsistent, is rejected.

The reference-flow module determines which merges can use the strict-match annotation, and the rights-management module performs strict merging as necessary. Upon receiving a new reference, the reference-flow module sends a merge request to the rights-management module. The merge request includes the IDs of the two asset IDs to merge and a Boolean flag indicating when strict-merge annotation can be used. The response includes a field to indicate when strict-match annotation has been used.

Asset consistency

Assets with multiple references that do not match each other are generally considered inconsistent; there is no easy way to determine which pieces of intellectual property such assets correspond to. Reference-based merges typically have two requirements: (a) a high match between two references, one from each of the two assets; and (b) a match between versions of the metadata of the assets. Although such an approach yields a high precision (99%+), it doesn't result in sufficient recall. To improve recall, the match-metadata requirement can be dropped, e.g., the metadata can be discarded if both references strictly match. Rejecting metadata can be better than forgoing merging, which creates split ownership, or leaves both metadata versions active, in turn making further metadata-based merging difficult.

However, if one of the two assets is inconsistent, then the risk of overmerging arises: two assets that should not be merged may get merged. To reduce the risk of overmerging, asset consistency is tested for before enabling strict-match based merges. The asset consistency signal is also useful to automatically measure the extant scale of overmerging, and can be used when developing automatic tools to unmerge overmerged assets .

The asset consistency signal is an attribute of an asset. It indicates whether all the active and inactive references linked to that asset strictly match each other. Determining the consistency of an asset generally requires: (a) knowing which references are linked to that asset; and (b) generating all the matches between those. The following can be used to compute the asset consistency signal.

- Determine the list of assets to update: Here, the list of references stored along the consistency signal are checked against the current list of references. If the two lists do not match, the signal is stale and is recomputed. At this point, the signal is checked to verify

that it is indeed marked stale. If not, the process that failed to mark the signal as stale is determined and corrected to ensure correct future operation. The decision to recompute the signal can also be based on the version of the signal, e.g., if the current version differs from the stored one, it is recomputed.

- Retrieve the existing matches between the pairs of references: Generating asset matches between all the pairs of references can be expensive, e.g., if an asset points to many (10,000+) references or if the references are long (20 hours or more). To limit the number of comparisons whenever possible, a database is utilized to store many of the matches that the match module produces. All matches for the requested references are first retrieved from the database. The missing matches are determined, and, as explained below, generated, and, including empty matches, added to the database.
- Generate the missing asset matches: Generating missing asset matches entails loading asset fingerprints from a fingerprint repository and performing a compare-operation for each pair of missing matches.
- Store the newly generated asset matches.
- Recompute the signal: At this point, all matches are known, and the consistency signal can be computed. As mentioned earlier, an asset can be declared consistent if all the references strictly match each other. The logic to compute the consistency signal can evolve; hence the consistency signal can be appended with a version-ID.
- Store the recomputed asset-consistency signal alongside the asset in a database.

As mentioned earlier, inconsistency in an asset is an indication of overmerging, and can be used to prevent further merging of other assets with that asset.

Using the consistency and the strict-match signals to merge assets

Upon upload, strict merges occur if a normal merge is infeasible. In general, an already overmerged asset (as determined by its consistency signal) is not further merged with another asset. On the other hand, a strict match between two assets with consistent signals results in a merge even if their metadata don't match. Merge-related procedures are executed within the reference-flow and the rights-management modules as follows.

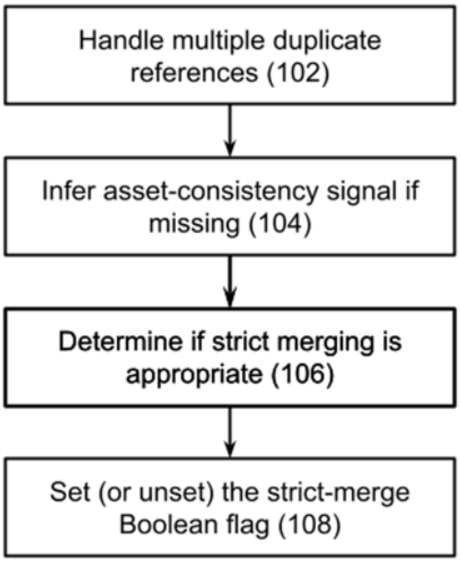

Fig. 1: Merge-related procedures within the reference-flow module

Fig. 1 illustrates merge-related procedures within the reference-flow module, explained in greater detail below.

- Process multiple duplicate references (102) In case of multiple duplicate references, if matching references are linked to other assets, merges are attempted until one succeeds or all fail. However, a merge is not attempted if the processed reference matches another reference from the asset it is linked to.
- Infer asset-consistency signal if missing (104) The asset consistency signal can be computed, for example, once a day. It is possible that a new asset temporarily lacks a

consistency signal. Alternatively, for older assets, a consistency signal can be temporarily stale before replacement by a recently computed consistency signal. The following heuristic is used to infer consistency for assets that lack a valid consistency signal: if only a single reference is linked to an asset, then that asset is consistent. This is because if just one reference is linked to the asset, then there can be no two linked references that are in conflict.

- Determine if strict merging is appropriate (106) Strict merging between two assets is allowed
    a. if both assets are consistent, as determined by their consistency signals, or, if they lack a consistency signal, by the above heuristic; and
    b. the strict parts of the match cover a substantial portion of both references.
- Set the strict-merge Boolean flag (108) As explained earlier, if strict merging is determined as being appropriate, the rights-flow module sends a merge request to the rights-management module with a Boolean flag indicating that strict-merge annotation can be used regardless of the outcome of metadata matching.

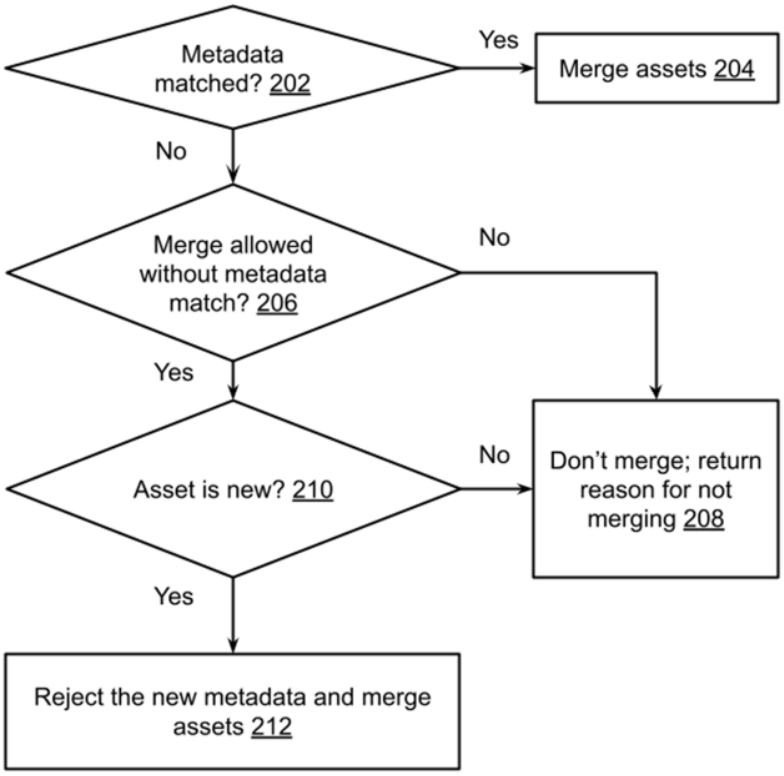

Fig. 2: Merge-related procedures within the rights-management module

Fig. 2 illustrates merge-related procedures within the rights-management module, explained in greater detail below. If the metadata of the two assets under consideration for merging match (202), then the assets are merged (204). If the metadata don't match, then the Boolean flag (received from the reference-flow module) is checked to see if merge is allowed without metadata match (206). If the Boolean flag is unset, then the assets are not merged, and the reason is returned for not merging (208). If the Boolean flag is set, then the asset is checked to determine if it is new (210). If so, its metadata is rejected and the assets are merged (212). If not, the assets remain unmerged, and the reason for not merging (208) is returned.

In this manner, the described techniques of asset consistency and strict-match annotation reconcile content assets (e.g., audio, video) in an asset-management database such that undermerging and overmerging is reduced. Slightly transformed duplicates, e.g. remasters,

trimmed versions, sped-up versions, etc., are readily detected and merged. The techniques enable the detection of overmerged and undermerged assets at scale, obviating substantial work that is otherwise necessary to be performed manually. The techniques can be utilized in any digital rights management situation, e.g., to reduce overmerging/undermerging for a database used to match user-contributed content or other content.

CONCLUSION

This disclosure describes techniques to automatically resolve overmerged assets and undermerged assets in a rights management database. Per the techniques, a consistency signal of an asset is computed. A logic module uses the consistency signal and a strict-match signal to merge assets that correspond to the same piece of intellectual property. Another logic module detects duplicate reference material and builds a graph of thus-far undermerged assets that can be merged. The techniques detect and resolve undermerged and overmerged assets at scale, correctly handling slightly transformed duplicates, e.g., remasters, trimmed versions, sped-up versions, etc.